School Connect Online

SCO INTERNATIONAL CODING OLYMPIAD

CLASS 10 SAMPLE QUESTION PAPER

A professional academic document for students, teachers, schools, and parents

Designed from Class 10 coding pathways and aligned with SCO's official Olympiad preparation, practice, reporting, and future-ready technology growth.

- Class 10 coding readiness guidance for secondary-level learners globally
- pathways across Python, SQL, C/C++, Swift, PHP, data analytics, security, APIs, and project studios
- exam structure, question practice, answer explanations, and technology-career readiness for academic enrichment

Python	SQL	C / C++	Swift	PHP Web
Data	Projects	Security	APIs	Debugging

Class 10 - Sample Question Paper - Set A

Paper Snapshot	
Exam Name	SCO International Coding Olympiad
Class / Grade	Class 10
Academic Year	
Question Paper Set	A
Duration	60 minutes
No. of Questions	50 objective-type questions
Core Areas	Advanced Programming, Application Development, Coding Projects, Data Analytics, Security, SQL, Swift, C, PHP, Python
Candidate Instructions	
Total Questions	50 objective-type questions
Duration	60 minutes
Marking	One correct answer for each question. Calculators are not allowed.
Response Rule	Select only one option for each question and review before final submission.

Question Paper with Answer Key and Explanations

Q.1

What will be the output of the following Python code?

```
   x = 5
   while x > 0:
x -= 1
   if x == 2:
break
   else:
   print("Loop Finished")
   print("End of Code")
```

A) Loop Finished

B) End of Code

C) Loop Finished End of Code

D) No Output

Answer: B) End of Code

Explanation: The else block in a while loop runs only if the loop completes normally. Since the break statement terminates the loop, the else block never executes.

Q.2

Consider the following SQL tables:

Customers

CustomerID | Name | City

101 | Alice | New York

102 | Bob | Boston

103 | Carol | Chicago

Orders

OrderID | CustomerID | Amount

1 | 101 | 250

2 | 103 | 300

3 | 101 | 400

What will the following SQL query return?

```
   SELECT Name, SUM(Amount)
   FROM Customers
JOIN Orders
   ON Customers.CustomerID = Orders.CustomerID
   GROUP BY Name;
```

A)

Alice, 650

Carol, 300

B)

Alice, 250

Carol, 300

Bob, NULL

C)

Alice, 650

Bob, 0

Carol, 300

D)

Alice, 650
Bob, NULL
Carol, 300

Answer: A) Alice, 650 | Carol, 300

Explanation: The JOIN only includes customers who have orders. Bob does not have an order, so he is excluded from the results.

Q.3

Which of the following PHP code snippets is the best way to handle a file upload securely?

A)

```
move_uploaded_file($_FILES['file']['tmp_name'], "uploads/" . $_FILES['file']['name']);
```

B)

```
$file = $_FILES['file']['tmp_name'];
if (mime_content_type($file) == "image/jpeg") {
move_uploaded_file($file, "uploads/" . $_FILES['file']['name']);
}
```

C)

```
$file = $_FILES['file']['tmp_name'];
if ($_FILES['file']['size'] < 1000000) {
move_uploaded_file($file, "uploads/" . $_FILES['file']['name']);
}
```

D)

```
if ($_FILES['file']['type'] == "image/png") {
move_uploaded_file($_FILES['file']['tmp_name'], "uploads/" . $_FILES['file']['name']);
}
```

Answer: B) Checking MIME type before moving the file.

Explanation: Option A is incorrect because it does not validate the file type, making it vulnerable to malicious uploads.
Option C is incorrect because it only checks size, not content type.
Option D is incorrect because $_FILES['file']['type'] can be spoofed easily.

Q.4

What will be the output of this C program?

```
#include <stdio.h>
int main() {
int arr[] = {10, 20, 30, 40};
int *ptr = arr;
printf("%d ", *(ptr + 2));
return 0;
}
```

A) 10
B) 20
C) 30
D) 40

Answer: C) 30

Explanation: ptr points to arr[0]. ptr + 2 moves the pointer to arr[2], which is 30.

Q.5

What will be the output of the following Swift code?

```
var str: String? = nil
if let value = str {
print(value)
} else {
print("No value")
}
```

A) nil
B) No value
C) Compilation error
D) Runtime error

Answer: B) No value

Explanation: Since str is nil, the if let statement does not execute the print(value), so it falls to the else block.

Q.6

What will be the output of this Python code?

```
x = [i for i in range(5) if i % 2 == 0]
print(x)
```

A) [0, 2, 4]
B) [1, 3, 5]
C) [0, 1, 2, 3, 4]
D) [2, 4]

Answer: A) [0, 2, 4]

Explanation: The list comprehension filters out numbers where i % 2 == 0, keeping only even numbers.

Q.7

What does the following Pandas code do?

```
df = df.dropna(subset=['Age'])
```

A) Removes rows where Age column has NaN values
B) Removes the entire Age column
C) Replaces NaN in Age with 0
D) Raises an error

Answer: A) Removes rows where Age column has NaN values.

Explanation: dropna(subset=['Age']) removes rows where the Age column has missing values.

Q.8

Which SQL query finds customers who never placed an order?

A)
```
SELECT Name FROM Customers WHERE CustomerID NOT IN (SELECT CustomerID FROM Orders);
```
B)
```
SELECT Name FROM Customers WHERE CustomerID IS NULL;
```
C)
```
SELECT Name FROM Customers WHERE CustomerID != Orders.CustomerID;
```
D)
```
SELECT Name FROM Customers JOIN Orders ON Customers.CustomerID = Orders.CustomerID WHERE Orders IS NULL;
```

Answer: A) Using NOT IN with a subquery.

Explanation: Option A is correct because it selects customers not in the Orders table.
Option B is incorrect because CustomerID is never NULL.
Option C is incorrect because it gives incorrect filtering.
Option D is incorrect because JOIN will not work without LEFT JOIN.

Q.9

Which SQL query is the most optimized way to find the total number of orders placed by each customer?

A)
```
SELECT CustomerID, COUNT(OrderID) FROM Orders;
```
B)
```
SELECT CustomerID, COUNT(OrderID) FROM Orders GROUP BY CustomerID;
```
C)
```
SELECT DISTINCT CustomerID, COUNT(OrderID) FROM Orders;
```
D)
```
SELECT CustomerID FROM Orders GROUP BY CustomerID;
```

Answer: B) Using GROUP BY CustomerID.

Explanation: Option A is incorrect because it does not group orders per customer.
Option C is incorrect because DISTINCT does not work correctly with COUNT().
Option D is incorrect because it does not return the count of orders.
Using GROUP BY CustomerID ensures we get the count of orders per customer efficiently.

Q.10

Machine Learning in Python - Linear Regression

What will happen if you train a linear regression model on highly correlated features?

A) The model will perform better
B) The model will suffer from multicollinearity
C) The model will give higher accuracy
D) The model will ignore correlated features automatically

Answer: B) The model will suffer from multicollinearity

Explanation: When features are highly correlated, it creates redundancy and unstable coefficient estimates.
Multicollinearity makes it difficult to determine the effect of each feature.
Solution: Use Principal Component Analysis (PCA) or remove highly correlated features.

Q.11

Which of the following statements about Swift Structs and Classes is true?

A) Structs and Classes both support inheritance
B) Structs are value types, while Classes are reference types
C) Structs can modify their properties without mutating keyword
D) Classes in Swift do not support deinitialization

Answer: B) Structs are value types, while Classes are reference types.

Explanation: Structs in Swift are value types, meaning they copy data when passed.
Classes are reference types, meaning they share data when passed.
Structs do not support inheritance (Option A is false).
Structs require mutating keyword to modify properties (Option C is false).
Classes support deinitialization (deinit) (Option D is false).

Q.12

What is the most effective way to prevent Cross-Site Request Forgery (CSRF) attacks in PHP?

A) Using HTTPS instead of HTTP

B) Storing passwords in plain text

C) Using CSRF tokens for form submissions

D) Relying on JavaScript validation only

Answer: C) Using CSRF tokens for form submissions.

Explanation: CSRF attacks trick users into performing unintended actions on authenticated sessions.
CSRF tokens are unique random values added to form submissions to verify requests.
Option A is incorrect because HTTPS encrypts data but does not prevent CSRF.
Option B is incorrect because storing passwords in plain text is a security risk.
Option D is incorrect because JavaScript validation can be bypassed easily.

Q.13

What will be the output of this C function pointer program?

```
#include <stdio.h>
void display(int x) {
printf("Value: %d", x);
}
int main() {
void (*funcPtr)(int) = display;
(*funcPtr)(10);
return 0;
}
```

A) Compilation Error

B) Value: 10

C) Runtime Error

D) Segmentation Fault

Answer: B) Value: 10

Explanation: funcPtr is a function pointer that stores the address of the function display.
(*funcPtr)(10) calls the function and prints Value: 10.
The program executes correctly without errors.

Q.14

Consider the following two Pandas DataFrames:

```
import pandas as pd
df1 = pd.DataFrame({'ID': [1, 2, 3], 'Name': ['Alice', 'Bob', 'Charlie']})
df2 = pd.DataFrame({'ID': [2, 3, 4], 'Score': [85, 90, 95]})
result = pd.merge(df1, df2, on='ID', how='inner')
print(result)
```

What will be the output?

A)

ID Name Score
0 2 Bob 85
1 3 Charlie 90

B)

ID Name Score
0 1 Alice NaN

1 2 Bob 85
2 3 Charlie 90
3 4 NaN 95

C)

ID Name Score
0 1 Alice 0
1 2 Bob 85
2 3 Charlie 90
3 4 NULL 95

D)

ID Name Score
0 1 Alice NULL
1 2 Bob 85
2 3 Charlie 90

Answer: **A) The matching rows for ID = 2 and 3 are merged, excluding ID = 1 and 4.**

Explanation: how='inner' performs an INNER JOIN, keeping only matching ID values.
ID 1 and 4 are excluded because they are not common in both DataFrames.

Q.15

Which statement about AJAX and Fetch API is correct?

A) AJAX and Fetch API are both based on XMLHttpRequest
B) Fetch API is promise-based, whereas AJAX uses callbacks
C) Fetch API does not support asynchronous requests
D) AJAX can work only with JSON responses

Answer: **B) Fetch API is promise-based, whereas AJAX uses callbacks.**

Explanation: AJAX (Asynchronous JavaScript and XML) relies on XMLHttpRequest, which requires callbacks.
Fetch API is newer, built on Promises, making code more readable.
Option A is incorrect because Fetch API does not use XMLHttpRequest.
Option C is incorrect because Fetch API supports async requests by default.
Option D is incorrect because AJAX can handle XML, JSON, and plain text responses.

Section - Application Development

Q.16 **Section - Application Development**

Which Python web framework is widely used for building scalable and high-performance web applications?

A) Flask
B) Django
C) Tornado
D) Pyramid

Answer: **B) Django**

Explanation: Django is a full-stack framework with built-in ORM, authentication, and security features.
Flask is lightweight but requires third-party extensions.
Tornado is used for real-time applications, not full-stack development.
Pyramid is modular but not as widely used as Django.

Q.17 **Section - Application Development**

Which SQL property ensures that either all changes occur, or none occur, during a database transaction?

A) Consistency
B) Isolation
C) Atomicity
D) Durability

Answer: C) Atomicity

Explanation: Atomicity ensures a transaction is treated as a single unit. If any part fails, the entire transaction is rolled back.
Consistency ensures the database remains valid after the transaction.
Isolation ensures transactions don't interfere with each other.
Durability guarantees that once committed, changes are permanent

Q.18 **Section - Application Development**

What will be the output of the following Swift code?

```
var name: String? = nil
print(name ?? "Unknown")
```

A) Unknown
B) nil
C) Compilation Error
D) Runtime Error

Answer: A) Unknown

Explanation: The ?? operator provides a default value when an optional is nil.
Since name is nil, "Unknown" is printed instead.

Q.19 **Section - Application Development**

Which of the following is the best way to prevent file upload vulnerabilities in PHP?

A) Store uploaded files in /uploads/ directory
B) Allow only .jpg, .png, and .pdf extensions
C) Check the MIME type of uploaded files
D) Validate file content using finfo_file()

Answer: D) Validate file content using finfo_file().

Explanation: Hackers can rename malicious files to .jpg, bypassing extension checks.
finfo_file() verifies the actual content type, preventing malware uploads.

Q.20 **Section - Application Development**

Which of the following is the most efficient way to filter rows in a Pandas DataFrame?

A) df[df["Age"] > 30]
B) df.loc[df["Age"] > 30]
C) df.query("Age > 30")
D) df[df.Age > 30]

Answer: C) df.query("Age > 30")

Explanation: query() is optimized for fast filtering, especially for large datasets.
df.loc[df["Age"] > 30] works but is slower.

Q.21 **Section - Application Development**

What will be the output of this C program?

```
#include <stdio.h>
int main() {
int arr[] = {10, 20, 30};
int *ptr = arr;
printf("%d", *(ptr + 2));
return 0;
}
```

A) 10
B) 20
C) 30
D) Compilation Error

Answer: C) 30

Explanation: ptr points to arr[0].
*(ptr + 2) moves 2 positions ahead, accessing arr[2] = 30.

Q.22 **Section - Application Development**

Which HTTP method is used to update a resource in a REST API?

A) GET
B) POST
C) PUT
D) DELETE

Answer: C) PUT

Explanation: PUT updates a resource.
POST creates a new resource.
GET retrieves data.
DELETE removes a resource.

Q.23 **Section - Application Development**

Which type of indexing improves performance for queries with multiple conditions?

A) Single-column index
B) Composite index
C) Unique index
D) Full-text index

Answer: B) Composite index

Explanation: A composite index covers multiple columns, making WHERE col1 = X AND col2 = Y faster.

Q.24 **Section - Application Development**

Which function is used to calculate the correlation between two columns in Pandas?

A) df.mean()
B) df.corr()
C) df.var()
D) df.describe()

Answer: B) df.corr()

Explanation: df.corr() computes Pearson correlation coefficients between columns.

Q.25 **Section - Application Development**

Which feature in Swift prevents memory leaks by deallocating objects automatically?

A) Garbage Collection
B) ARC (Automatic Reference Counting)
C) Manual Memory Management
D) Swift does not handle memory

Answer: B) ARC (Automatic Reference Counting)

Explanation: ARC automatically deallocates unused objects, preventing memory leaks.

Q.26 **Section - Application Development**

Which is the best way to prevent SQL Injection in PHP?

A) Escape input using addslashes()
B) Use prepared statements with PDO
C) Use mysql_real_escape_string()
D) Limit user input length

Answer: B) Use prepared statements with PDO.

Explanation: Prepared Statements prevent SQL injection by binding parameters safely.

Q.27 **Section - Application Development**

Which Python library is best for deep learning?

A) NumPy
B) Pandas
C) TensorFlow
D) Matplotlib

Answer: C) TensorFlow

Explanation: TensorFlow is an AI and deep learning library by Google.

Q.28 **Section - Application Development**

Which function is used to deallocate memory in C?

A) malloc()
B) calloc()
C) free()
D) realloc()

Answer: C) free()

Explanation: free() releases dynamically allocated memory.

Q.29 **Section - Application Development**

Which normal form removes transitive dependencies?

A) 1NF
B) 2NF
C) 3NF
D) BCNF

Answer: C) 3NF

Explanation: 3NF eliminates transitive dependencies, making the schema more efficient.

Q.30 **Section - Application Development**

What is the most secure way to store session data?

A) Store in browser cookies

B) Store in a database with encrypted tokens

C) Store in local storage

D) Store in JavaScript variables

Answer: B) Store in a database with encrypted tokens

Explanation: Encrypted session tokens stored in a database improve security against session hijacking.

Section - Coding Projects

Q.31 **Section - Coding Projects**

You are developing a user authentication system for a web application. The system should securely store user passwords and prevent SQL Injection attacks.

Which method is the best approach to store passwords securely in PHP?

A) Store passwords as plain text in the database

B) Hash passwords using md5() before storing them

C) Use password_hash() with PASSWORD_BCRYPT

D) Encrypt passwords with base64_encode() before storing

Answer: C) Use password_hash() with PASSWORD_BCRYPT

Explanation: Plain text passwords (Option A) are a major security risk.

md5() (Option B) is outdated and vulnerable to rainbow table attacks.

base64_encode() (Option D) is not encryption and can be reversed easily.

Best practice: Use password_hash($password, PASSWORD_BCRYPT), which applies salting and strong hashing to prevent attacks.

Q.32 **Section - Coding Projects**

You are designing a social media database where users can have multiple posts and multiple followers. The system should quickly retrieve posts made by a user's followers.

Which database design approach would optimize query performance?

A) Store posts and followers in the same table

B) Use a Many-to-Many relationship with an index on user_id

C) Store all follower relationships in a single JSON column

D) Query all posts from the entire database and filter in PHP

Answer: B) Use a Many-to-Many relationship with an index on user_id

Explanation: Option A: Storing posts and followers in the same table creates data redundancy.

Option C: Using a JSON column is inefficient for large datasets.

Option D: Querying all posts without an index is slow and increases server load.

Best approach: Use a Many-to-Many relationship (Followers table) and index user_id to retrieve posts efficiently.

Q.33 **Section - Coding Projects**

You are working on a machine learning project in Python to predict house prices based on historical data. Your dataset has missing values in several columns.

Which approach is the best way to handle missing data in Pandas?

A) Remove rows with missing values using df.dropna()

B) Replace missing values with the mean using df.fillna(df.mean())

C) Fill missing values with 0

D) Ignore missing values and train the model as-is

Answer: B) Replace missing values with the mean using df.fillna(df.mean())

Explanation: Dropping rows (dropna()) (Option A) reduces data, affecting model accuracy.
Filling missing values with 0 (Option C) may mislead the model.
Ignoring missing values (Option D) causes errors in training.
Best practice: Use df.fillna(df.mean()) for numerical data to maintain accuracy.

Q.34 Section - Coding Projects

You are developing an iOS fitness tracking app in Swift. The app needs to store workout logs locally and synchronize them with a cloud database when the internet is available.

Which is the best storage solution for offline data?

A) Store logs in UserDefaults

B) Save logs in a .txt file

C) Use Core Data for structured storage

D) Store logs in a remote database only

Answer: C) Use Core Data for structured storage

Explanation: UserDefaults (Option A) is not meant for large data storage.
Text files (.txt) (Option B) lack querying capabilities.
Remote database only (Option D) fails offline.
Best approach: Use Core Data, which provides structured local storage, supports queries, and allows easy synchronization with a cloud database.

Q.35 Section - Coding Projects

You are designing a real-time traffic light system using C programming and an embedded microcontroller. The traffic lights should operate based on timers and sensor input to detect vehicles.

Which control structure is most efficient for continuously checking sensor input and updating light states?

A) if-else inside main()

B) while(1) loop with conditional checks

C) for(;;) loop with delay() function

D) Use goto statements for jumping between light states

Answer: B) while(1) loop with conditional checks

Explanation: if-else inside main() (Option A) runs only once, not continuously.
for(;;) loop (Option C) with delay() wastes processing time.
goto statements (Option D) make code hard to maintain.
Best approach: while(1) runs infinitely, efficiently checking sensor inputs and updating traffic lights in real time.

Q.36 Section - Coding Projects

Scenario:

You are developing an e-commerce website that allows users to submit product reviews. However, a security audit has revealed that attackers can inject malicious JavaScript into review submissions, leading to Cross-Site Scripting (XSS) attacks.

Question:

Which approach is the most effective way to prevent XSS vulnerabilities in user-submitted data?

A) Use htmlspecialchars() in PHP to escape special characters

B) Store user reviews in plain text without validation

C) Allow JavaScript code inside reviews for flexibility

D) Use eval() to execute user-submitted scripts in a sandbox

Answer: A) Use htmlspecialchars() in PHP to escape special characters

Explanation: Option B (Plain text storage) allows XSS risks.
Option C (Allowing JavaScript code) makes the site vulnerable to XSS.
Option D (eval()) is a security risk.
Best practice: Use htmlspecialchars($input, ENT_QUOTES, 'UTF-8') to sanitize user input and prevent script execution.

Q.37 Section - Coding Projects

You are building an AI-powered chatbot for a customer support system. The chatbot should process user queries, respond appropriately, and learn from interactions using Machine Learning (ML).

Question:

Which Natural Language Processing (NLP) technique is the best for making the chatbot understand context in conversations?

A) Use TF-IDF (Term Frequency-Inverse Document Frequency) to find important words
B) Implement Bag-of-Words (BoW) to count word occurrences
C) Use Recurrent Neural Networks (RNNs) or Transformers for context-based learning
D) Hard-code all possible user inputs and responses

Answer: C) Use Recurrent Neural Networks (RNNs) or Transformers for context-based learning

Explanation: Option A (TF-IDF) only ranks words but does not understand context.
Option B (BoW) counts words but does not consider word order.
Option D (Hardcoding responses) is impractical for dynamic conversations.
Best approach: Use RNNs (like LSTMs) or Transformers (like GPT or BERT) for context-aware chatbot responses.

Q.38 Section - Coding Projects

Scenario:

You are working on a fraud detection system for an online payment platform. Your dataset contains transaction records, and you need to identify suspicious transactions using SQL queries and Python-based Machine Learning models.

Which SQL query would best identify unusual transactions that are significantly larger than a user's average spending?

A)
```
SELECT user_id, amount
FROM transactions
WHERE amount > (SELECT AVG(amount) FROM transactions);
```
B)
```
SELECT user_id, amount
FROM transactions
WHERE amount > 2 * (SELECT AVG(amount) FROM transactions WHERE user_id = transactions.user_id);
```
C)
```
SELECT user_id, amount
FROM transactions
WHERE amount > 10000;
```
D)
```
SELECT user_id, amount
FROM transactions
ORDER BY amount DESC;
```

Answer: B) Identify transactions that exceed twice the user's average spending

Explanation: Option A checks transactions above the global average, missing user-specific fraud.
Option C only flags transactions above 10000, ignoring relative spending patterns.
Option D orders transactions but does not detect anomalies.
Best approach: Option B dynamically detects anomalies by checking if a transaction is twice the user's average spending.

Q.39 **Section - Coding Projects**

You are developing a fitness tracking app in Swift for iOS. The app should track step count, calories burned, and heart rate using HealthKit API and store user progress.

Question:

Which Swift method would be best to continuously track step count while preserving battery life?

A) Use a while(true) loop to fetch step count every second

B) Use CoreMotion's CMPedometer to fetch step count updates

C) Constantly query HealthKit every second for step data

D) Run a background task that requests data every minute

Answer: B) Use CoreMotion's CMPedometer to fetch step count updates

Explanation: Option A (while(true) loop) drains the battery.

Option C (Querying HealthKit every second) is inefficient.

Option D (Background task every minute) can miss steps.

Best approach: Use CMPedometer().startUpdates() to track steps efficiently.

Q.40 **Section - Coding Projects**

You are developing an embedded system in C programming for a smart traffic light that adjusts signal duration based on real-time traffic density. The system should use IR sensors to detect vehicle count.

Question:

Which control structure is best for continuously checking sensor input and adjusting light duration?

A) Use a simple if-else condition inside main()

B) Use an infinite while(1) loop with sensor checks and time delays

C) Use a switch-case statement to handle light states

D) Run a for-loop that cycles through all sensor inputs once

Answer: B) Use an infinite while(1) loop with sensor checks and time delays

Explanation: Option A (if-else inside main()) runs only once.

Option C (switch-case) does not handle continuous sensor input well.

Option D (for-loop) only checks inputs once instead of continuously.

Best approach: while(1) loop continuously checks sensor input and adjusts traffic signals dynamically.

Q.41 **Section - Coding Projects**

Real-time Data Processing - Live Stock Price Updates

Scenario:

You are developing a stock trading platform in Python that uses real-time data to display live stock prices. The system uses WebSocket to listen for stock price updates and display them on the user interface.

Question:

Which of the following Python features is most suitable for implementing the live data updates and ensuring that the GUI remains responsive while handling WebSocket data?

A) Use a single-threaded while loop that continuously checks for updates and updates the UI

B) Use asynchronous programming with asyncio to manage WebSocket connections without blocking the UI thread

C) Implement a polling system where the UI checks for updates every few seconds

D) Use multiprocessing to handle the WebSocket data and update the UI in separate processes

Answer: B) Use asynchronous programming with asyncio

Explanation: Option A (Single-threaded loop) will block the UI thread, making it unresponsive.

Option C (Polling) is inefficient and may result in delays.

Option D (Multiprocessing) is not ideal for a GUI-based application that requires fast updates.

Best approach: asyncio allows you to handle real-time data updates without blocking the main thread, ensuring a responsive user interface.

Q.42 **Section - Coding Projects**

Advanced SQL Query Optimization for Large Datasets

Scenario:

You are developing a web-based inventory management system using SQL. The database contains millions of records, and queries are becoming slow. You need to optimize a query to calculate the total value of all items in the inventory, considering their prices and quantities.

Question:

Which of the following methods would most improve performance for the query that calculates the total value of items in the inventory?

A) Use nested subqueries in the SELECT statement to filter data

B) Create indexes on the columns price and quantity to speed up data retrieval

C) Use LEFT JOINs instead of INNER JOINs to include all items

D) Use GROUP BY to combine all the results into one record before performing the calculation

Answer: B) Create indexes on the columns price and quantity

Explanation: Option A (Nested subqueries) may actually slow down the query, as they involve additional operations.

Option C (LEFT JOIN) can result in extra records and is not needed when we want only matched rows.

Option D (GROUP BY) is useful for grouping, but it doesn't directly impact query performance.

Best practice: Indexes on frequently queried columns, like price and quantity, can drastically improve data retrieval speed, especially in large datasets.

Q.43 **Section - Coding Projects**

Swift Programming - App Development with User Authentication

Scenario:

You are building a fitness tracking app in Swift that requires user authentication using email and password. You need to securely handle user login and registration in the app using Firebase Authentication.

Question:

Which of the following Swift methods would be most suitable for securely registering a user and authenticating them using Firebase?

A) Use Auth.auth().signIn(withEmail:password:) for both login and registration

B) Use Auth.auth().createUser(withEmail:password:) for registration and signIn(withEmail:password:) for login

C) Use signIn(withEmail:) and implement manual password validation

D) Store the password locally and check it against a hardcoded value for authentication

Answer: B) Use Auth.auth().createUser(withEmail:password:) for registration and signIn(withEmail:password:) for login

Explanation: Option A is incorrect because the signIn method is only for logging in, not for registration.

Option C involves insecure manual password validation, which is not recommended.

Option D is unsafe, as it requires storing passwords locally.

Best practice: Use Firebase's built-in createUser for registration and signIn for login to ensure secure user authentication.

Q.44 **Section - Coding Projects**

C Programming - Handling Complex Data with Structures

Scenario:

You are building an application in C to manage a student database. Each student has a name, ID number, grades, and attendance record. You need to create a program that can dynamically store and retrieve information about students using structures.

Question:

Which of the following C constructs will best allow you to store and manipulate a dynamic array of student records efficiently?

A) Use a static array of fixed size and overwrite old data when new data is added

B) Use dynamic memory allocation with malloc to create a flexible array of student records

C) Store each student's data in a global variable to maintain persistent state

D) Use linked lists to store student records as they are more efficient than arrays

Answer: B) Use dynamic memory allocation with malloc

Explanation: Option A (Static array) is inefficient and won't allow for dynamic changes in the number of students.

Option C (Global variable) can lead to difficult-to-manage code and poor maintainability.

Option D (Linked lists) could work but is overcomplicating the task when arrays are sufficient.

Best approach: Dynamic memory allocation with malloc allows you to create an array of student records with a size that can change at runtime, making the code more flexible.

Q.45 Section - Coding Projects

Advanced Python - Predictive Model for Sales Forecasting

Scenario:

You are developing a predictive sales forecasting model using Python and machine learning. The dataset contains sales data for the past 5 years, and you want to build a model that predicts sales for the upcoming months using regression.

Question:

Which of the following Python libraries is most commonly used for building a linear regression model to predict sales and should be preferred for this task?

A) Pandas for data manipulation and cleaning

B) Matplotlib for plotting data visualizations

C) Scikit-learn for implementing the regression model

D) TensorFlow for deep learning neural networks

Answer: C) Scikit-learn for implementing the regression model

Explanation: Option A (Pandas) is great for data manipulation but not for regression modeling.

Option B (Matplotlib) is used for visualizations, not for building models.

Option D (TensorFlow) is more suitable for deep learning rather than simpler tasks like linear regression.

Best practice: Scikit-learn provides easy-to-use tools for implementing linear regression models and other common machine learning algorithms.

Q.46 Section - Coding Projects

Debugging C Program - Array Index Out of Bounds

Scenario:

You are given a C program that attempts to calculate the average of numbers stored in an array. However, there is a bug in the code that results in an array index out of bounds error.

Question:

What is the error in the following C program and how can it be fixed?

```
#include <stdio.h>
int main() {
int arr[5] = {10, 20, 30, 40, 50};
int sum = 0, i;
for (i = 0; i <= 5; i++) {  // Loop is going out of bounds
sum += arr[i];
}
printf("Average: %f", sum / 5);
```

```
    return 0;
}
```

A) The program accesses index 5, which is out of bounds for an array of size 5.

B) The loop should start from i = 1 instead of i = 0.

C) The calculation of the average is incorrect.

D) The program does not handle division by zero.

Answer: A) The program accesses index 5, which is out of bounds for an array of size 5.

Explanation: In the for loop, the condition i <= 5 causes the loop to run until i = 5, which is out of bounds since array indices in C start from 0.

The fix is to change the loop condition to i < 5.

Q.47 Section - Coding Projects

Debugging Python Program - Incorrect String Formatting

Scenario:

You are given a Python program that reads a list of numbers and prints each number with a specified formatting. However, it produces an error when run.

Question:

What is the error in the following Python code, and how can it be fixed?

```
numbers = [1, 2, 3, 4, 5]
for num in numbers:
print("Number: {num:05d}")
```

A) There is a syntax error in the print statement due to incorrect string formatting.

B) The {num:05d} formatting is not valid for integers.

C) The variable num should be wrapped in parentheses in the print statement.

D) The code will print Number: {num:05d} literally instead of formatting.

Answer: D) The code will print Number: {num:05d} literally instead of formatting.

Explanation: In Python, to use f-string formatting, the correct syntax is f"Number: {num:05d}".

Without the f before the string, the placeholder {num:05d} is treated as part of the literal string and will not perform formatting.

Fix: print(f"Number: {num:05d}")

Q.48 Section - Coding Projects

Debugging SQL Query - Incorrect Join

Scenario:

You are tasked with writing an SQL query to retrieve customer details along with their order amounts from two tables: customers and orders. However, the query does not return the expected result.

Question:

What is the issue with the following SQL query?

```
SELECT customers.name, orders.amount
FROM customers
LEFT JOIN orders ON customers.id = orders.customer_id
WHERE orders.amount > 100;
```

A) The LEFT JOIN should be replaced with an INNER JOIN.

B) The WHERE clause should not filter out NULL values from the orders table.

C) The query is correct and returns the expected result.

D) The WHERE clause is applied after the join and filters out NULL values, making it behave like an INNER JOIN.

Answer: D) The WHERE clause is applied after the join and filters out NULL values, making it behave like an INNER JOIN.

Explanation: In an SQL query, the WHERE clause is applied after the JOIN operation.

Since the LEFT JOIN includes all rows from the customers table and the orders table with NULL values for customers with no orders, filtering by orders.amount > 100 will exclude these NULL values and turn the query into an INNER JOIN.
To fix this, move the condition on orders.amount to the ON clause or use IS NOT NULL in the WHERE clause to preserve the LEFT JOIN behavior.

Q.49 **Section - Coding Projects**

Scenario:

You are working with a C++ program that dynamically allocates memory for a large array but is causing memory leaks.

Question:

What is causing the memory leak in the following C++ code, and how can it be fixed?

```
#include <iostream>
int main() {
int* arr = new int[100];  // Memory allocation
arr[0] = 10;
arr[1] = 20;
// Forgot to delete the dynamically allocated memory
return 0;
}
```

A) The new operator is not required in C++ for dynamic memory allocation.

B) Memory allocated using new must be deallocated using delete[].

C) The program is correct and does not have any memory leaks.

D) The program should use smart pointers instead of manual memory management.

Answer: **B) Memory allocated using new must be deallocated using delete[].**

Explanation: In C++, memory allocated using new must be explicitly deallocated using delete[] to avoid memory leaks.
The program does not call delete[] arr, so the allocated memory is not freed, leading to a memory leak.
Fix: delete[] arr; after using the allocated memory.

School Connect Online

Answer Key

Question / Answer	Question / Answer	Question / Answer	Question / Answer	Question / Answer
Q.1 - -	Q.1 - B	Q.2 - A	Q.3 - B	Q.4 - C
Q.5 - B	Q.6 - A	Q.7 - A	Q.8 - A	Q.9 - B
Q.10 - B	Q.11 - B	Q.12 - C	Q.13 - B	Q.14 - A
Q.15 - B	Q.16 - B	Q.17 - C	Q.18 - A	Q.19 - D
Q.20 - C	Q.21 - C	Q.22 - C	Q.23 - B	Q.24 - B
Q.25 - B	Q.26 - B	Q.27 - C	Q.28 - C	Q.29 - C
Q.30 - B	Q.31 - C	Q.32 - B	Q.33 - B	Q.34 - C
Q.35 - B	Q.36 - A	Q.37 - C	Q.38 - B	Q.39 - B
Q.40 - B	Q.41 - B	Q.42 - B	Q.43 - B	Q.44 - B
Q.45 - C	Q.46 - A	Q.47 - D	Q.48 - D	Q.49 - B

School Connect Online

Space for Rough Work